Digital Design LU

Lab Exercise 1

Jakob Lechner, Thomas Polzer
{lechner, tpolzer}@ecs.tuwien.ac.at
Department of Computer Engineering
University of Technology Vienna

Vienna, October 4, 2011

1 Overview

The first lab exercise will make you acquainted with the tools used in this lab course for implementing FPGA designs. A basic FPGA design flow consists of a simulator and a synthesis tool. The simulator is used for verifying and debugging the functionality and the timing of the circuit implemented in a high-level hardware description language (VHDL in our case). The synthesis tools translates the behavioral and/or structural description into a gate-level netlist. This netlist can then be mapped to the FPGA's logic cells. Finally the produced bitstream file is used to configure the FPGA.

2 Required Reading

- Digital Design LU – Design Flow Tutorial
- Digital Design VO – Der Logikanalysator

3 Task Description

In this exercise you will have to simulate and synthesize the VHDL description of a memory controller, which performs a series of write accesses for storing data into an SRAM chip. After the first half of the write accesses is completed the controller is idle for approximately 2 seconds. Then the second half is transmitted. Once all write operations have been finished, the SRAM controller waits another 2 seconds and then starts over again with the first character.

In the source directory for this exercise you can find two subdirectories named *asram* and *ssram*. Both directories contain a working VHDL description of the mentioned memory controller. The controller in the *asram*-directory is able to perfom write accesses on *asynchronous* SRAM chips, whereas the controller in the *ssram*-directory is able to perfom write accesses on *synchronous* SRAM chips. Table 1 shows the output signals of both memory interfaces.

Table 1: SRAM memory signals.

Signal Name	Asynchronous	Synchronous
Clk		x
Address	x	x
Data	x	x
Chip Select (CS)	x	x
Write Enable (WE)	x	x

Before the SRAM chip can store new data, obviously the desired address and data value need to be applied to the address and data bus, respectively. The write access can then be triggered by forcing the CS and WE signals low[1] for a certain period of time. The semantics of the four control signals (address, data, CS, WE) are basically the same for both synchronous and asynchronous SRAMs. However, the timing requirements are a litte bit different, as can be seen in the waveforms in Figure 1 and Figure 2.

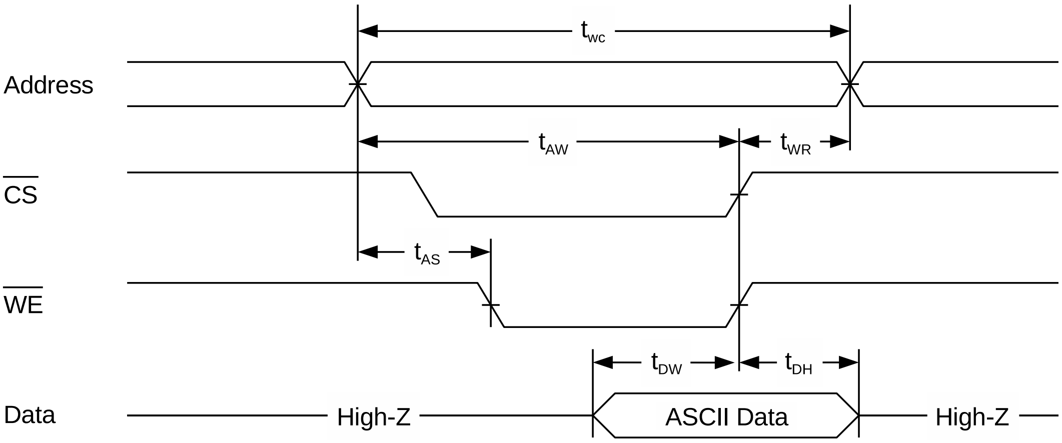

Figure 1: Timing diagram of an asynchronous write access.

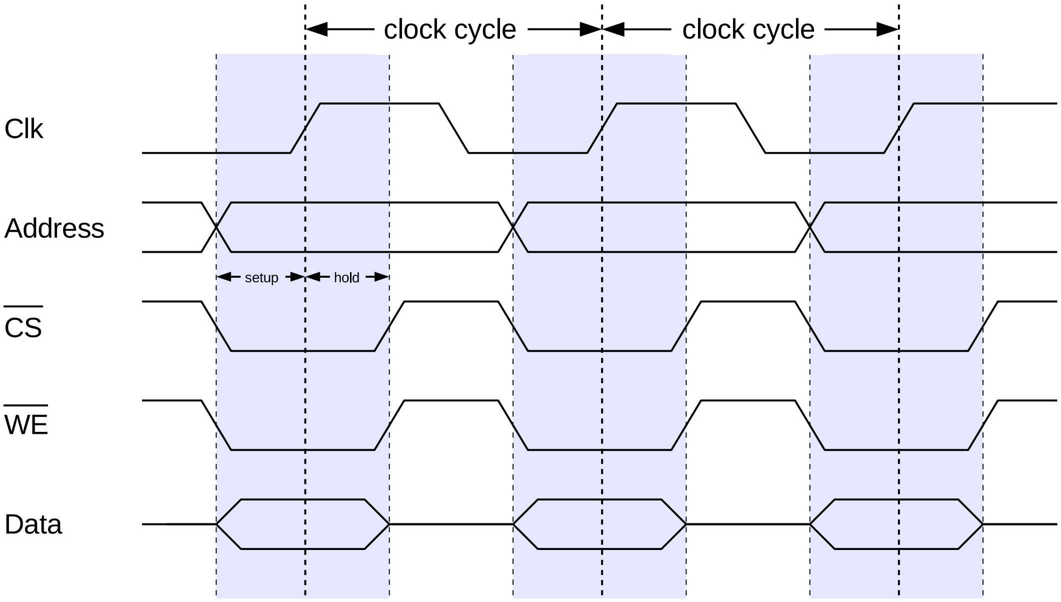

Figure 2: Timing diagram of a synchronous write access.

---

[1] CS and WE are low-active

In case of the asynchronous SRAM there is no clock signal for coordinating the timing of the control signals. Therefore the datasheet of an asynchronous SRAM chip specifies a range of timing requirements between the control signals, as can be seen in Figure 1. In comparison, the timing in case of the synchronous SRAM interface is rather simple. All control signals need to be set a certain time before the rising clock edge (setup time) and remain stable for a certain time after the clock edge (hold time). In Figure 2 this setup/hold window is shown as a gray box around the rising clock edge.

Task 1: ASRAM – Behavioral Simulation Perform a behavioral simulation of the *asynchronous* SRAM controller. An appropriate testbench is provided in the simulation directory. Add all signals of the top-level entity to the waveform window and run the simulation long enough for tracing all write accesses before the mentioned idle interval. Take screenshots showing a trace of all characters of the transmitted string in readable form and include these screenshots in your lab protocol.

Furthermore zoom in to one of the write accesses and measure all timing intervals marked in Figure 1 with the help of markers. Take a screenshot showing the marker pair for one of the timing parameters. Compare the measured timings to the minimum/maximum requirements for write accesses in the datasheet of the K6RK6R4016V1D-10 SRAM chip (the datasheet can be found on the course website). Does the SRAM controller comply to all requirements? If not, which timings are violated?

Task 2: ASRAM – Synthesis, Place&Route Create a new Quartus project for a Cyclone IV E EP4CE115F29C7 FPGA and add the VHDL files of the *asynchronous* memory controller (except the testbench file of course). Assign the pins of the input and output ports accordingly. The FPGA pin names for clock and reset inputs can be found in the pinout description on the course website. To be able to measure the SRAM signals assign the output pins of the SRAM interface (address, data, cs, we) to the GPIO extension connector of the FPGA board.

Task 3: ASRAM – Post-layout Simulation Use the netlist file (.vho) and the timing file (.sdo), which were generated during the previous task, for performing a post-layout simulation[2]. The testbench file used in the behavioral simulation can also be employed for post-layout simulation.

---

[2] Depending on the settings, the Quartus timing analyzer might produce two sets of vho and sdo files: one with fast and one with slow timings. For this exercise use the conservative (slow) timing estimates.

The timing file provides information on the real physical signal delays. Therefore, signals do not switch instantaneously after the clock edge, in constrast to a behavioral simulation. Every single bit of a signal vector switches individually depending on the propagation and routing delays of the corresponding circuitry. Run the simulation long enough in order to take a screenshot of some transition on the address bus of the memory interface. Zoom into the waveform until you can see the diffenrent delays of the address signal and use two markers to measure the duration between the first and the last bit toggling.

Task 4: ASRAM – Logic Analyzer Measurments Configure the FPGA by downloading the bitstream file (.sof) generated during compilation. Attach the logic analyzer probes to the extension connector of FPGA board accordingly to the used pin mappings. Do not forget to attach the ground connector of the probes to a ground pin on the FPGA board. Configure the logic analyzer appropriately for tracing all signals of the SRAM interface. Devise a trigger condition which starts the measurement exactly after the idle interval in order to record the second part of the write accesses. Make screenshots where the transferred ASCII characters can be seen and include them in your lab protocol. Which operation mode of the logic analyzer do you need to use for tracing the signals of the asynchronous memory controller – state mode or timing mode?

Furthermore sample the trace of a single write access with the highest resolution possible and use markers to determine the timing parameters that can be seen in Figure 1. Verifiy if the values are equal to the ones you measured during the behavioral simulation.

Finally, try to capture a waveform for measuring the length of the idle timeout between the first and the second part of the write accesses. Take a screenshot and insert the measured value into your protocol.

Task 5: SSRAM – Logic Analyzer Measurments Create a new Quartus project for the *synchronous* memory controller. You can use the settings from the Quartus project of the *asynchronous* memory controller (just copy the existing tcl script and create the Quartus project in the *ssram* directory). Map the additional clock output signal to an approriate pin on the FPGA board's extension connector and compile the design. Download the sof-file on the FPGA and connect the clock signal to a probe of the logic analyzer. Now trace the whole string written to the SRAM and take screenshots where the individual characters can be seen. Which operation mode of the logic analyzer do you need to use for tracing the signals of the synchronous memory controller – state mode or timing mode?

4 Submission

Deadline: October 19^{th}, 2011, 23:59

Upload a zip archive containing:

- PDF file of the lab protocol